BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
Please do not report the images to the
Image Problem Mailbox.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 02-273867
(43)Date of publication of application : 08.11.1990

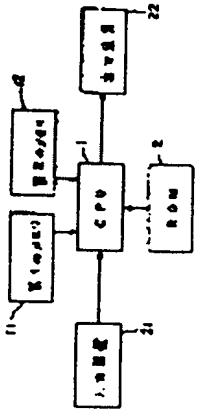
(51)Int.Cl. G06F 15/347

(21)Application number : 01-096079
(22)Date of filing : 14.04.1989

(71)Applicant : SHARP CORP
(72)Inventor : AKAHA BANE TOSHIO
FUJIMOTO KOJI
FUKUDA NAOYUKI

(54) MATRIX ARITHMETIC UNIT

(57)Abstract:
PURPOSE: To reduce the storage quantity and the calculation quantity of a memory by storing data for specifying a zero element or a non-zero element with regard to each element of a matrix, and referring to it at the time of operation.
CONSTITUTION: This unit is provided with a first memory 11 for storing data for specifying a zero element or a non-zero element with regard to each element of a matrix shown by a two-dimensional arrangement, a second memory 12 for storing data for showing the contents of the non-zero element of the matrix, and a discriminating means 1 for discriminating whether the element of the matrix is zero or not by referring to the data stored in a first memory 11. In this state, with regard to the element of the matrix which is discriminated not to be zero by the discriminating means 1, the sum of products is derived by multiplying the data stored in a second memory 12 and an element of an input vector. In such a way, in the case of executing an arithmetic processing of a sparse matrix of a large scale, the storage quantity and the calculation quantity of a memory are reduced by a ratio occupied by a non-zero element in a transformation matrix.



⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平2−273867

⑤Int.Cl.⁵	識別記号	庁内整理番号	㊸公開　平成2年(1990)11月8日
G 06 F 15/347		P 7056−5B	

審査請求　未請求　請求項の数 2　(全7頁)

㊹発明の名称　行列演算装置

㉑特　願　平1−96079
㉒出　願　平1(1989)4月14日

㉗発明者　赤羽　俊夫　大阪府大阪市阿倍野区長池町22番22号　シャープ株式会社内
㉗発明者　藤本　好司　大阪府大阪市阿倍野区長池町22番22号　シャープ株式会社内
㉗発明者　福田　尚行　大阪府大阪市阿倍野区長池町22番22号　シャープ株式会社内
㉑出願人　シャープ株式会社　大阪府大阪市阿倍野区長池町22番22号
㉔代理人　弁理士　青山　葆　外1名

明　細　書

1．発明の名称

行列演算装置

2．特許請求の範囲

(1)　2次元配列で表わされる行列の各要素について零である要素か零でない要素かを特定するデータを格納する第1のメモリと、

上記行列の零でない要素の内容を表わすデータを格納する第2のメモリと、

上記第1のメモリに格納されたデータを参照して、上記行列の要素が零であるか否かを判別する判別手段と、

上記判別手段によって零でないと判別された行列の要素について、上記第2のメモリに格納されたデータと入力ベクトルの要素とを乗算して、積和を求める演算手段を備えたことを特徴とする行列演算装置。

(2)　上記第1のメモリは、零である要素が連続して並ぶ数を表わす整数によって零である要素を特定するデータを格納することを特徴とする請求項1に記載の行列演算装置。

3．発明の詳細な説明

＜産業上の利用分野＞

この発明は、零成分を多く含む行列(スパース行列)とベクトルとの演算に適した行列演算装置に関する。

＜従来の技術＞

自然界における現象を計算機を用いてシュミレーションする場合、2次元配列で表わされる行列を変換行列として、1次元配列で表わされるベクトルの1次変換を計算することが多い。例えば、次式(1),(2)に示すような行列演算が挙げられる。

$$y = Wx \qquad \cdots(1)$$

$$v^t = u^t W \qquad \cdots(2)$$

ここで、$x=(x_1,x_2,x_3,\cdots,x_N)^t$ は入力ベクトル、$y=(y_1,y_2,y_3,\cdots,y_N)^t$ は出力ベクトル、$W=(Wji)$は一次変換のためのN行M列の変換行列、$u=(u_1,u_2,u_3,\cdots,u_N)^t$ は入力ベクトル、$v=(v1,v2,v3,\cdots,v_N)^t$ は出力ベクトル、$(*)^t$ は行と列を入れ換えた転置行列を示している。

従来のベクトルプロセッサなどの行列演算装置は、上記変換行列Wの各要素Wjiを表わすデータを記憶するメモリと、この要素Wjiと入力ベクトルの要素との積和の計算アルゴリズムを記憶する記憶手段と、この計算アルゴリズムに従って計算する演算手段とを備えて、(1)式の計算のとき出力yj(j=1,2,…,N)を、次式(3)に従って計算するようにしている。

$$y_j = \sum_{i=1}^{M} W_{ji} x_i \qquad \cdots(3)$$

また(2)式の計算のとき各列の出力 v_i(i=1,2,…,M)を、次式(4)に従って計算するようにしている。

$$v_i = \sum_{j=1}^{N} W_{ji} u_j \qquad \cdots(4)$$

なお、これら式(3),(4)の計算を模式的に示すとそれぞれ第16図、第17図のようになる。

<発明が解決しようとする課題>

ところで、上記行列演算を現実の問題に適用するにあたって、上記変換行列Wの要素Wjiのうち零である要素(以下、「零要素」と呼ぶ)の占める割合が大きくなる場合がある。たとえば、神経回路網のシュミレーションにおいて、一方の神経回路素子群が他方の神経回路素子群から受け取る伝達信号は、送り手の各素子の出力を入力ベクトルxとし、送り手側の各素子から受け手側の各素子への結合の強さ(結合係数)を変換行列Wとした一次変換y=Wxと考えることができるが、このとき、すべての神経回路素子間が接続されていることは稀であって、逆に、各素子間の結合係数すなわち変換行列Wの要素のうち大部分が零要素である(スパース行列である)場合が多い。この傾向は神経回路網が大規模になるほど強くなる。

このような場合、上記従来の演算処理装置は、零でない要素(以下、「非零要素」と呼ぶ)が多い行列を取り扱う場合と同様に、上記N行M列の変換行列Wの各要素Wjiをそのまま N×M個の実数としてメモリに割り当てて記憶する必要があり、また、上記行列演算処理1回につき乗算と加算とをN×M回ずつ行なっている。このため、零要素を記憶・計算する無駄が生じていると考えられる。

そこで、この発明の目的は、大規模なスパース行列演算処理を行なうときにメモリの記憶量と計算量を低減することができる演算処理装置を提供することにある。

<課題を解決するための手段>

上記目的を達成するために、この発明の演算処理装置は、2次元配列で表わされる行列の各要素について零要素か非零要素かを特定するデータを格納する第1のメモリと、上記行列の非零要素の内容を表わすデータを格納する第2のメモリと、上記第1のメモリに格納されたデータを参照して、上記行列の要素が零であるか否かを判別する判別手段と、上記判別手段によって零でないと判別された行列の要素について、上記第2のメモリに格納されたデータと入力ベクトルの要素とを乗算して、積和を求める演算手段を備えたことを特徴としている。

また、上記第1のメモリは、零要素が連続して並ぶ数を表わす整数によって零要素を特定するデータを格納するのが望ましい。

<作用>

上記判別手段によって第1のメモリに格納されたデータを参照して、参照した上記行列の要素が零であるときは、何ら計算を行なうことなく、次の要素の参照を続ける。そして、参照した要素が零でないとき、上記演算手段によって上記第2のメモリに格納されたデータと、入力ベクトルのこのデータに対応する要素とを乗算する。1つの行または列について、この積和を計算して、出力ベクトルの1つの要素とする。そして、各行または各列について、この計算を行なって、出力ベクトルの全要素を求める。

このように行列演算処理を行なう場合、例えば上記行列の全要素(N×M個の実数)のうち非零要素の占める割合がk%であるとき、この行列の要素を記憶するための上記第2のメモリの記憶量は、実数にしてN×M×k/100個分となる。また、上記行列演算処理1回につき乗算と加算を行なう回数は、それぞれN×M×k/100となる。したがっ

て、非零要素の占める割合が少ない(kが小さい)ときに、上記行列の要素の記憶量と上記演算処理の計算量が低減される。

また、上記第1のメモリは、零要素が連続して並ぶ数を表わす整数によって零要素を特定するデータを格納する場合、零要素か非零要素かの判断回数が少なくて済み、上記第1のメモリの記憶量は、整数にして約N×M×k／100個分となる。したがって、このkが小さいときに、上記行列の各要素について零または非零を特定するための記憶量が低減される。

＜実施例＞

以下、この発明の行列演算装置を図示の実施例により詳細に説明する。

第1図はこの発明の第1の実施例を示している。この行列演算装置は、ＣＰＵ(中央演算処理装置)1と、所定の計算アルゴリズムを記憶するＲＯＭ2と、変換行列Wについての情報を記憶する第1のメモリ11および第2のメモリ12と、入力ベクトル$x=(x_1,\cdots,x_j,\cdots,x_N)^t$または$u=(u_1,\cdots,u_j,\cdots u_N)^t$の情報を入力する入力装置21と、出力ベクトル$y=(y_1,\cdots,y_j,\cdots y_N)^t$または$v=(v_1,\cdots,v_j,\cdots,v_M)^t$の情報を出力する出力装置22を備えている。

上記ＣＰＵ1は、上記入力装置21から入力ベクトルxの各要素を表わすデータを受けて、上記第1のメモリ11および第2のメモリ12を参照し、ＲＯＭ2が記憶する計算アルゴリズムに従って、上記入力ベクトルxまたはuの一次変換を計算して、出力ベクトルyまたはvを表わすデータを上記出力装置22に出力することができる。第7図に示すように、上記入力装置21は、入力ベクトルxの各要素xjを表わすデータを保持可能な入力バッファ302およびこの入力バッファ302の各データＸＴxp(xp＝1,2,…,M)を指すポインタ(指示値xp)306と、入力ベクトルuの各要素ujを表わすデータを保持可能な入力バッファ304およびこの入力バッファ304の各データＵＴup(up＝1,2,…N)を指すポインタ(指示値up)308とからなっている。上記出力装置22は、出力ベクトルyの各要素yjを表わすデータを保持可能な積和演算バッファ兼用の出力バッファア303およびこの出力バッファ303の各データＹＴyp(yp＝1,2,…,N)を指すポインタ(指示値yp)307と、出力ベクトルvの各要素vjを表わすデータを保持可能な積和演算バッファ兼用の出力バッファア305およびこの出力バッファ305の各データＶＴvp(vp＝1,2,…M)を指すポインタ(指示値vp)309とからなっている。なお、第7図中の301は、この演算処理装置の機能を説明するために、例として変換行列Wの各要素Wjiを2次元配列によって表わしたものである。図中、"0"はWji＝0である零要素、"W"はWji≠0である非零要素を表わしている。また、piは零要素が行方向に並ぶ数、qiは非零要素が行方向に並ぶ数を表わしている。第2図に示すように、上記第1のメモリ11は、上記変換行列Wの零要素が連続して並ぶ数を表わす整数を記憶しているインデックステーブル401と、このインデックステーブル401の各データＩＴip(ip＝1,2,…)を指すポインタ(指示値ip)403とからなっている。一方、第3図に示すように、上記第2のメモリ12は、上記変換行列Wの非零要素の内容を表わすデータを順に格納している係数メモリ402と、この係数メモリ402の各データＷＴwp(wp＝1,2,…)を指すポインタ(指示値wp)404とからなっている。

上記インデックステーブル401、係数メモリ402は、次のようにして作成される。第7図に示した上記変換行列W301の各行を1行目から順に左から右に調べてゆき、非零要素のときその内容(実数)を表わすデータを、上記係数メモリ402に格納する一方、この非零要素の左側に並ぶ零要素の数piに1を足した整数(pi＋1)をnビットのデータで表わして上記インデックステーブル401に格納する(以下、単に「整数を登録する」という)。なお、上記非零要素の左隣が非零要素である場合、pi＝0であるため、登録する整数は1となる。非零要素がqi個並ぶときは上記インデックステーブル401には整数1を(qi－1)個続けて登録することになる。各行の行末にきたときは、

行末記号delim(delim＝2　－1)を登録する。行末が零要素である場合、この行末の零要素を含む零要素の並びの数(零要素が並んでおらず、左隣が非零要素のときは1)を登録するのでなく、行末記号delimを登録する。ところで、このようにnビットのデータ(1ワード)で整数を表わす場合、表わすことができる整数は(2　－1)までであり、さらに整数(2　－1)を上に述べたように行末記号delimに使用しているので、結局、1ワードで表すことができる整数は(2　－2)までとなっている。そこで、(2　－2)個以上零要素が並ぶときは、次のように2ワード以上使ってその数を表わして登録する。例えば、零要素が並ぶ数をpiとすると、

$$pi+1=(2^{n}-2)a+b \qquad a,bは整数$$
$$0 \leqq a \qquad 0 \leqq b<(2^{n}-2)$$

と表わせるときは、(a＋1)個のワードを使って表わす。すなわち、a個のワードのデータは$(2^{n}-2)$とし、最後の1ワードのデータはbとする。

この行列演算装置は、上記述べたように、変換行列Wの零要素が並ぶ数piと行末記号delimをインデックスとして、次のように演算処理を行なう。

入力ベクトルxの一次変換として式(1)を計算をする場合、第8図に示す計算アルゴリズムに従って計算する。

まず、ステップS_1に示すように、各ポインク403,404,306,307の指示値をそれぞれip,wp,yp＝1、xp＝0とし、出力バッファ303のデータYTyp(yp＝1,…,M)を0とする(初期化)。次に、インデックステーブル401のデータITipが行末記号delim(＝2　－1)であるかどうか判別(S_2)して、行末であれば改行(S_3)する。行末でなければ、行方向向きにITip分だけ移動(S_4)して、ITipが最大数(2　－2)であるかどうかを判別(S_5)する。最大数であれば、インデックステーブル401の次のデータを調べにゆく(S_7)。最大数でなければ、係WTwp×XTxpをYTypに加算(S_6)し、係数メモリ404の次のデータを出せるように指示値wpを1つ進めると共に、インデックステーブル401の次のデータを調べにゆく(S_8)。そして、ステップS_2に戻って、再びITipが行末記号delimであるかどうかを判別して、行末であれば改行(S_3)して、さらに、N行まで調べ終わったとき、この演算を終了する。

入力ベクトルu^{t}の一次変換式(2)を計算する場合、上記演算と同様の手順によって、第9図に示す計算アルゴリズムに従って計算する。

このように演算処理を行なうことによって、例えばN行M列の変換行列Wの全要素(N×M個の実数)のうち非零要素の占める割合がk%であるとき、この行列Wの要素を記憶するための上記係数メモリ402の記憶量は、実数にしてN×M×k／100個分となり、一方、上記インデックステーブル401の記憶量は、整数にして約N×M×k／100個分となる。したがって、非零要素の占める割合が少ない(kが小さいとき)上記変換行列Wの要素の記憶量を低減することができる。また、上記行列演算処理1回につき乗算と加算を行なう回数はそれぞれN×M×k／100回となって、kが小さいとき計算量を低減することができる。

次に、第2の実施例を説明する。

この演算処理装置は、第1の実施例のインデックステーブル401に代えて、第4図に示すインデックステーブル411を備えている。他の構成は第1の実施例と同一である。上記インデックステーブル411は次のようにして作成される。インデックステーブル401と同様に、零要素の並びの数piに1を足した整数(pi＋1)を登録する。ただし、行末記号delimを使用せず、零要素が行末から次行の行頭へ続く場合は、行末の零要素の並び数と次行の行頭の零要素の並び数とを足した数に1を加えて登録する。例えば、第7図に示す変換行列W301の1行目の行末と2行目の行頭の場合、整数(p_1+p_2+1)を登録する。

上記入力ベクトルx,入力ベクトルuの一次変換式(1),式(2)を計算する場合、それぞれ第10図、第11図に示す計算アルゴリズムに従って行なう。なお、簡単のため、各データ、指示値は第1の実施例と同一記号を使用している(後に述べ

る第３、第４の実施例において同様)。第１の実施例に対して略同一手順であるが、式(１)の計算の　合、行末を検出するためにxpとＭとを比較して、xp＞Ｍならば行が変わったと判断(S_{10})して、vpをint(xp／Ｍ)だけ進める(S_{10})点が異なっている。式(２)の計算の場合、vpを使ってこれを行なう。なお、int(＊)は括弧内の式の値の整数部を示している。

次に、第３の実施例を説明する。

この演算処理装置は、第１の実施例のインデックステーブル４０１に代えて、第５図に示すインデックステーブル４２１を備えている。他の構成は第１の実施例と同一である。上記インデックステーブル４２１は、零要素の並びの数piと別に非零要素の並びの数qiを登録する。すなわち、非零要素が並んでいる場合、第１の実施例，第２の実施例と異なり、(qi－１)個の整数１をそれぞれ別個に登録するのでなく、１つのデータとして整数qiを登録する。そして、１ワード当たりnビットのうち最上位ビットを、零要素の並びの数piであるか非零要素の並びの数qiであるかの区別に使用する。零要素または非零要素が行末から次行の行頭へ続くときは、それらの並びの数を足した整数(p_i+p_{i+1})，(q_i+q_{i+1})を登録する。

上記入力ベクトルx、入力ベクトルu^tの一次変換として式(１)，(２)を計算する場合、それぞれ第１２図，第１３図に示す計算アルゴリズムに従って演算処理を行なう。第１の実施例および第２の実施例に対して略同一手順であるが、ＩＴipが零要素または非零要素のいずれを示しているかを判断(S_{10}，S_{11})して、零要素を示しているときは、その数だけxpまたはvpをスキップする点が異なっている(S_{20}，S_{21})。非零要素を示しているときは、その数だけ入力ＸＴxpと係数ＷＴwpとの積和を計算する(S_{27}乃至S_{31}，S_{28}乃至S_{32})。ただし、第２の実施例と同様に、その途中で行末になったかどうかを、xpまたはvpの値をＭの値と比較して判断する(S_{30}，S_{33})。

次に、第４の実施例を説明する。

この演算処理装置は、第１の実施例のインデックステーブルに代えて、第６図に示すインデックステーブル４３１を備えている。他の構成は第１の実施例と同一である。上記インデックステーブル４３１は、第３の実施例と同様に、零要素の並びの数piと別に非零要素の並びの数qiを登録する。ただし、行末では零要素または非零要素の並びの数のいずれかの最大値を行末記号delimとして登録する。なお、行末が零要素または零要素の並びで終わるときは、１または並びの数を登録せず、上記行末記号delimを登録する。

上記入力ベクトルx，入力ベクトルutの一次変換式(１)，式(２')を計算する場合、それぞれ第１４図，第１５図に示す計算アルゴリズムに従って演算処理を行なう。第３の実施例に対して、行末であるかどうかを行末記号delimを使用して判断(S_{21}，S_{30})する点のみが異なっている。

なお、第１乃至第４の実施例において、変換行列Ｗの各行を左から右へスキャンしたが、当然ながら、列方向にスキャンしても良い。

＜発明の効果＞

以上より明らかなように、この発明の演算処理装置は、２次元配列で表わされる行列の各要素について零要素か非零要素かを特定するデータを格納する第１のメモリと、上記行列の非零要素の内容を表わすデータを格納する第２のメモリと、上記第１のメモリに格納されたデータを参照して、上記行列の要素が零であるか否かを判別する判別手段と、上記判別手段によって零でないと判別された行列の要素について、上記第２のメモリに格納されたデータと入力ベクトルの要素とを演算して、積和を求める演算手段を備えているので、大規模なスパース行列の演算処理を行なう場合、変換行列において非零要素の占める割合がk%であるとき、メモリの記憶量と計算量をk%に低減することができる。

また、上記第１のメモリは、零である要素が連続して並ぶ数を表わす整数によって零要素を特定するデータを格納するようにした場合、変換行列の各要素について零要素か非零要素か特定する回数をk%に低減することができ、第１のメモリの

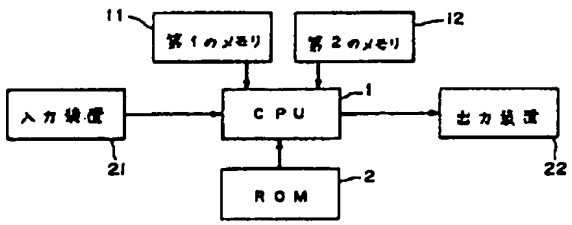
記憶量をk%に低減することができる。

4．図面の簡単な説明

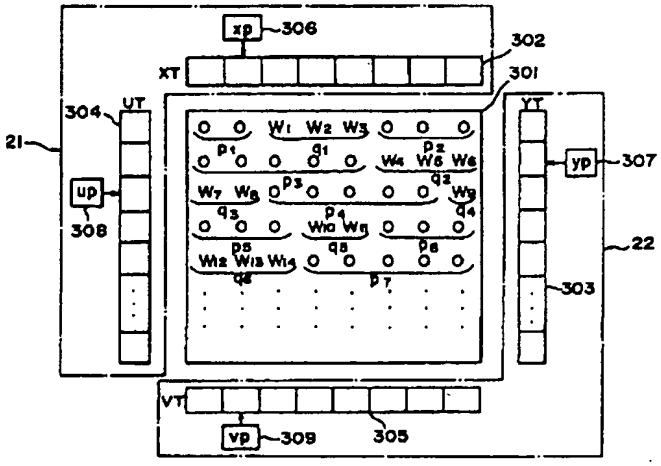
第1図はこの発明の演算処理装置の構成を示すブロック図、第2図，第4図，第5図および第6図は上記演算処理装置のインデックステーブルを示す図、第3図は上記演算処理装置の係数メモリを示す図、第7図は上記演算処理装置の入出力バッファ，ポインタと変換行列Wの要素を示す図、第8図乃至第15図は上記演算処理装置の計算アルゴリズムを示すフローチャート、第16図および第17図は従来の演算処理装置による演算を模式的に示す図である。

1…CPU、2…ROM、11…第1のメモリ、
12…第2のメモリ、21…入力装置、
22…出力装置、301…変換行列W、
302,308…入力バッファ、
303,305…出力バッファ、
401,411,421,431…インデックス　テーブル、
402…係数メモリ、
306,307,308,309,403,404…ポインタ。

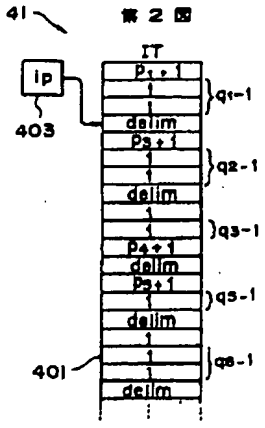
第1図



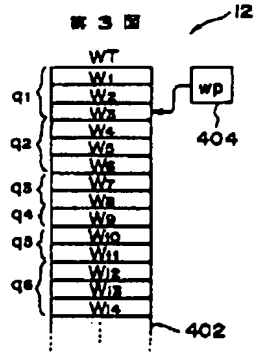
第7図



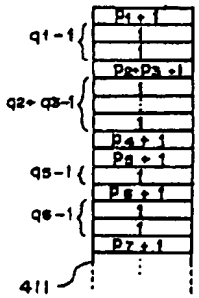
第2図



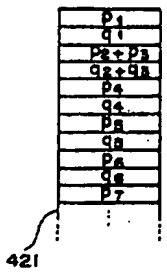
第3図



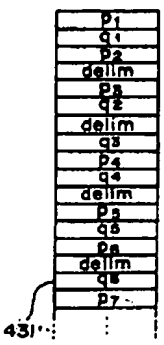
第4図



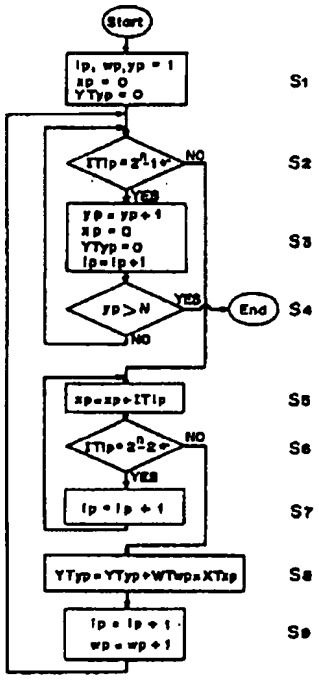
第5図



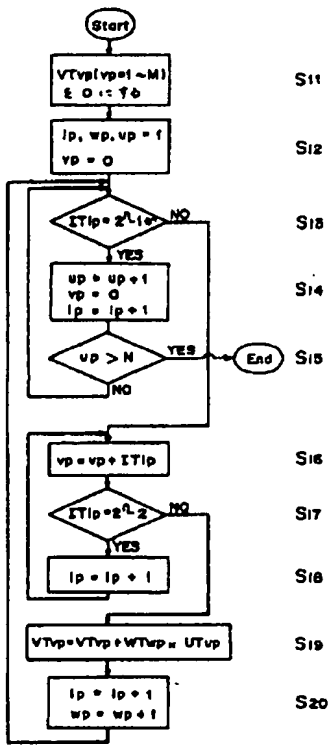
第6図



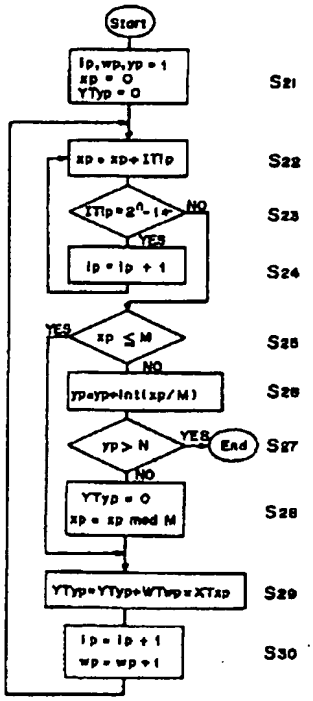
第8図



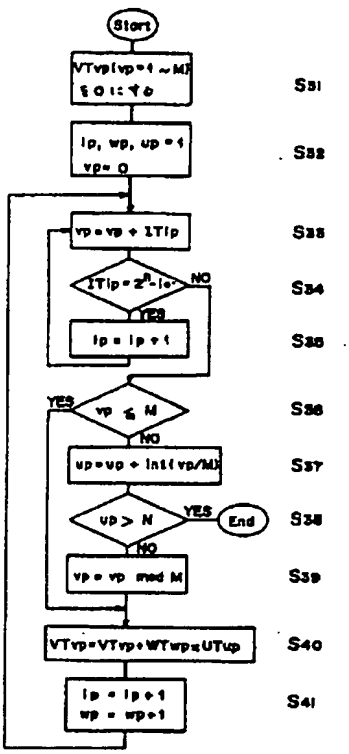
第9図



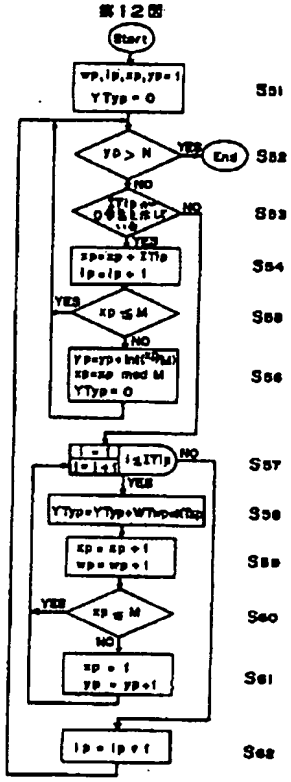
第10図



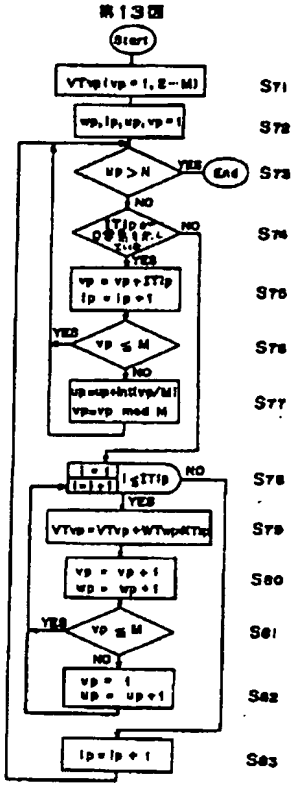
第11図



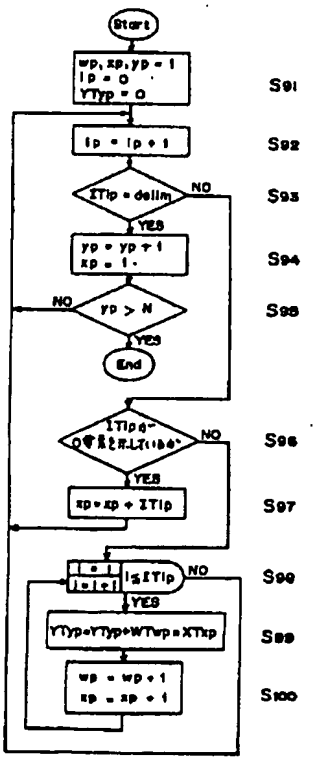
第12図



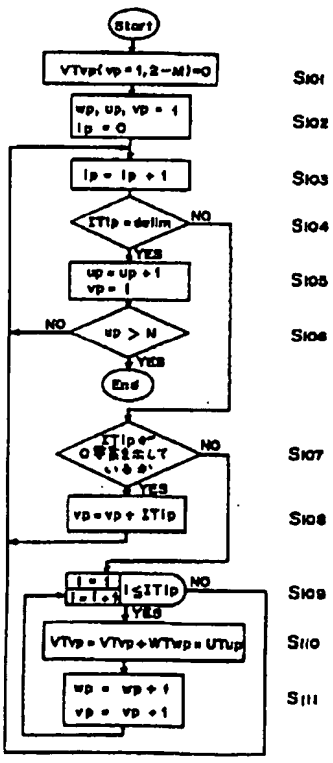
第13図



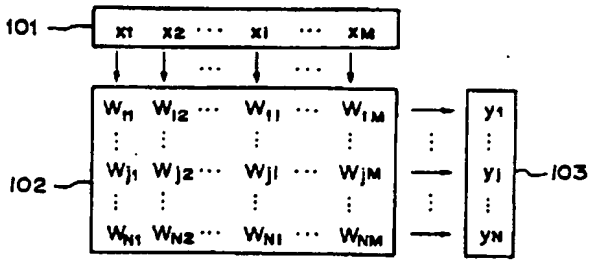
第14図



第15図



第16図



第17図